# Ex. No: 5 Hamming Code.

Aim:

To write a program to implement error detection and correction using hamming code concept.

Error correction at Data link layer:

Hamming code is a set of error-correction codes that can be used to detect and correct the errors that can occur when the data is transmitted from the sender to the receiver.

Create sender program with below features.

1) Input to sender file should be a text of any length. Program should convert the text to binary.

2) Apply hamming code concept on the binary data and add redundant bits to it.

3) Save this output in a file called channel.

Create a receiver program with below features.

1) Receiver program should read the input from channel file.

2) Apply hamming code on the binary data to check for errors

3) If there is an error, display the position of the error.

4) Else remove the redundant bits and convert the binary data to ascii and display the output.

Student observation:

```
def calc_parity_positions(m):
    r=0
    while (2**r)<(m+r+1):
        r+=1
    return r

def generate_hamming_code(data):
    m=len(data)
    r=calc_parity_positions(m)
    n=m+r
    hamming=['0']*(n+1)

    j=0
    for i in range(1,n+1):
        if i&(i-1)==0:
            continue
        hamming[i]=data[j]
        j+=1
```

```
for i in range(r):
    pos = 2**i
    parity = 0
    for j in range(1, n+1):
        if j & pos and j != pos:
            parity ^= int(hamming[j])
    hamming[pos] = str(parity)
return ''.join(hamming[1:])

def detect_and_correct(hamming_code):
    n = len(hamming_code)
    hamming = ['1','0'] + list(hamming_code)
    r = calc_parity_positions(n - bin(n).count
    error_pos = 0
    for i in range(r):
        pos = 2**i
        parity = 0
        for j in range(1, n+1):
            if j & pos:
                parity ^= int(hamming[j])
        if parity != 0:
            error_pos += pos
    if error_pos != 0:
        print(f"Error detected at bit
            position: {error_pos}")
        hamming[error_pos] = '1' if
        hamming[error_pos] == '0' else '0'
    else:
        print("No error detected")
    return ''.join(hamming[1:])
```

```
data = "1011"
print("Original data:", data)
hamming_code = generate_hamming_code(data)
print("Hamming Encoded data:", hamming_code)

error_pos = 3
hamming_with_error = list(hamming_code)
hamming_with_error[error_pos-1] = '1' if
hamming_with_error[error_pos-1] == '0'
else '0'
hamming_with_error = ''.join(hamming_with_
error)
print("Hamming code with Error:", hamming_
with_error)
corrected_code = detect_and_correct(hamming_
with_error)
print("Corrected hamming Code:", corrected_code)
```

Output:

```
Original Data: 1011
Hamming Encoded data: 0110011
Hamming Code with Error: 0100011
Error detected at bit position: 3
Corrected Hamming Code: 0110011
```

Result:

The program to implement hamming code is implemented successfully